SpaceWire on Earth Orbiting Scatterometers

Alex Bachmann, Minh Lang, James Lux, Richard Steffke
Jet Propulsion Laboratory
California Institute of Technology
4800 Oak Grove Drive
Pasadena, CA 91109
(818) 354-5087, (818) 354-1556, (818) 354-2075, (818) 354-3533
Bachmann@jpl.nasa.gov,Minh.Lang@jpl.nasa.gov, James.p.Lux@jpl.nasa.gov, Richard.Steffke@jpl.nasa.gov

Abstract-The need for a high speed, reliable and easy to implement communication link has led to the development of a space flight oriented version of IEEE 1355 called SpaceWire. SpaceWire is based on high-speed (200 Mbps) serial point-to-point links using Low Voltage Differential Signaling (LVDS). SpaceWire has provisions for routing messages between a large network of processors, using wormhole routing for low overhead and latency. [Additionally, there are available space qualified hybrids, which provide the Link layer to the user's bus]. A test bed of multiple digital signal processor breadboards, demonstrating the ability to meet signal processing requirements for an orbiting scatterometer has been implemented using three Astrium MCM-DSPs, each breadboard consists of a Multi Chip Module (MCM) that combines a space qualified Digital Signal Processor and peripherals, including IEEE-1355 links. With the addition of appropriate physical layer interfaces and software on the DSP, the SpaceWire link is used to communicate between processors on the test bed, e.g. sending timing references, commands, status, and science data among the processors. Results are presented on development issues surrounding the use of Spacewire in this environment, from physical layer implementation (cables, connectors, LVDS drivers) to diagnostic tools, driver firmware, and development methodology. The tools, methods, and hardware, software challenges and preliminary performance are investigated and discussed.

TABLE OF CONTENTS

1. INTRODUCTION

The need to send data reliably and quickly from subsystem to subsystem on spacecraft has led to various data transfer implementations. Serial methods are popular as they reduce the wire and pin count.

Spacewire is a high-speed serial communication link standard, which set specifications for the various layers. Its progenitor is IEEE1355 the commercial communication link, which is very prolific. SpaceWire adds LVDS and specifies different space qualified cable and connectors. We have developed a testbed for the next generation orbiting scatterometer [1][2] which uses the Astrium Multi Chip Module containing theTSC21020 DSP, Digital peripheral controller and a SpaceWire (the SMCS332) asic. Rather than attempt to implement some other inter processor link, we used the IEEE-1355/SpaceWire link as the communication method between the DSP processors.
Alternative technologies used in the past have strong heritage but technical or implementation limitations reduced or excluded them as an option on the testbed.

2. SPACEWIRE STANDARD

The main purpose of the SpaceWire standard is to ensure compatibility between different equipment, in order to facilitate integration, testing, reduce time-to-market, cost and re-use in different missions [3].

SpaceWire (proposed as IEEE 1355.2) is based on the IEEE 1355-1995 standard; it is a full-duplex, bi-directional, serial, point-to-point data link with a raw data rate of 200Mbit/sec. It encodes data using two differential signal pairs in each direction. A total of eight signal wires, four in each direction [4]. Each pair is twisted, shields of the four individual twisted shielded pairs are insulated from one another, and from the overall shield. encompassing the whole bundle.
Connections are made with 9 pin micro D-type connectors, which have strong flight legacy. The signaling method is LVDS.

LVDS

SpaceWire, allowing communication over distances of 10 m or more, uses Low Voltage Differential Signaling (LVDS). LVDS are current mode switched devices and require a good match between the characteristic impedance of the transfer medium (i.e. 100 to 100 ohm twisted pair) and the shunt termination resistance at the receiver end. Given the low voltages involved (350mV) requires more attention be paid to grounding, common mode rejection ratio (cmrr) and

[1] 0-7803-7231-X/01/$10.00/© 2002 IEEE

EMC design. This means balanced signal skew between the differential signals needs to be maintained by keeping the line lengths of similar distance. However the "constant current loop" signaling also helps reduce radiated noise and as a side benefit allows detection of opens or shorts in the cable.

Each link consists of two pairs of lines per direction for a total of 4 pairs or 8 individual wires. One of the differential pairs carries the data the other carries a strobe signal. XOR'd these two signals generate the clock. The strobe signal is dependent on the data, when the data remains constant from one data bit interval to the next a strobe transition occurs. This type of coding is called Data-Strobe (DS) encoding. This coding stream is diagramed below.

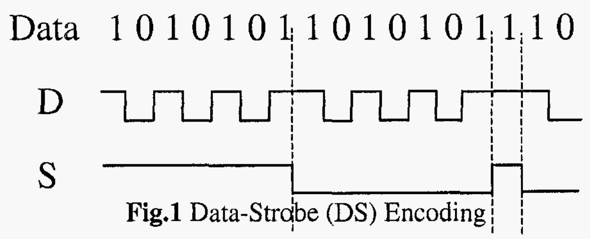


Fig.1 Data-Strobe (DS) Encoding

Two of the advantages to this type of coding are:

1. Since the resulting data is clocked on both the positive and negative edges of the recovered clock, one gets twice the bit rate as versus the more traditional leading edge clock with synchronous protocol.
2. And since the strobe signal gets switched only when the data bit was the same as the previous data bit, the di/dt type noise introduced into the system is much lower.

Any communication technology is broken down into levels which are characterized per a specification. The SpaceWire levels are described below.

Various SpaceWire Protocol Levels

- Level 0: Physical Level
- Level 1: Signal Level
- Level 2: Character Level
- Level 3: Exchange Level
- Level 4: Packet Level
- Level 5: Transaction Level

Physical Level—describes the physical interface between nodes in a link. This includes mechanical and electrical specifications, covering items like connectors and cabling.

Signal Level—describes the make up of the electrical signals going across a link; including voltage amplitudes, encoding (in this case DS) see Figure 1 above, fault protocol, and data rates

Character Level—describes how a basic character is defined using bits to build the basic unit of a character. Spacewire defines a control character and a data character. The control characters are made up of a parity bit, a data control flag (set to one) and a two-bit control code. These in different combinations indicate the following and are used to facilitate the exchange level.

- Flow Control (FCT)
- Normal End of Packet (EOP)
- Error End of Packet (EEP)
- Escape (ESC)
- Null (ESC + FCT)

The data character contains a parity bit, the data control flag bit (set to zero) and eight bits of data transmitted least significant bit first.

Exchange Level—describes how the data will be sent across the link. It contains the format or steps necessary to establish and maintain communication between any two nodes. This level contains the following elements.

- Initialization
- Flow Control
- Detection of Parity Errors
- Detection of Disconnect Errors
- Link Error Recovery

KNOWN

Initialization starts by placing both nodes in a know state, then checking for errors and followed by the sending out of NULL's, ending with the transmission of Data, FCT and Nulls as required.

Flow control characters control the availability of each node in a link. They indicate whether each side of node has space available to accept new incoming data thus avoiding overflow and loss of data. They are sent every n characters (8 in this case). Once received they are not stored in the receive memory or buffer. Thus a transmitter will not send any characters until it has received one or more FCT's to indicate that the receiver is ready. The transmitter keeps a credit count of the number of characters it has been authorized to send. Each time the receiver sends an FCT the transmitter increments its credit counter by 8. When it transmits a character it decrements its credit counter by 1. If the credit counter reaches zero the transmitter will stop sending characters until it receives another FCT. The NULL character is used to maintain the connectivity between two nodes. In order to remain connected, the nodes at both ends of a link must send NULL characters continuously when they do not have data to send.

Detection of Parity/Disconnect errors will result in the transmitted data to be erroneous or even missing. Therefore when SpaceWire detects one of these it will initiate a response accordingly. A disconnect error is registered when the time interval from the last transition on either the data or strobe signal exceeds the disconnect-detection time. If an error is detected SpaceWire follows the Link Error Recovery sequence.

Link Error Recovery starts with one node of the link ceasing transmission. This will cause a disconnect at the other node. This node then also stops transmitting. This generates a disconnect error at the original node. This procedure is known as "exchange of silence". Once this occurs both nodes cycle through the reset sequence (ErrorReset, ErrorWait, Ready) ending up in the Ready state ready to begin operation. If a error continues the working node will attempt to establish the link by sending NULL's. If it does not receive a response due to the error then the above cycle is repeated.

Packet Level—is a sequence of characters in a specific order and format. (in this case Hdr—Data—EOP/EEP) The Header dictates the destination of the packet. Data is the relevant information. Data can be any number of characters. And the EOP/EEP indicates the end of the packet.

Transaction Level—describes how a number of packets are assembled to perform a task or function.

These levels define and describe the elements necessary to build a SpaceWire link from the lowest level to the highest. Other protocols can be embedded in a SpaceWire packet such as MPEG, IP, ATM. This would be done at the transaction level.

3. Astrium Multi Chip Module

The Astrium Multi Chip Module (MCMDSP) is a radiation tolerant hybrid that incorporates a TSC21020F Digital Signal Processor, the SMCS332, and a Digital Peripheral Controller, which allows various peripherals to be connected to it making it a possible contender for a turnkey DSP system requiring limited hardware and software design man-hours. Fig. 2 shows the layout of the evaluation board containing the hybrid chip. Some of the hybrid features are:

- Peripheral Controller ASIC
- IEEE 1355 (SpaceWire) High Speed 150Mbit/s serial links
- JTAG
- User Interface UART, FIFO, I/O port
- Radiation tolerant to 50krads(Si) dose
- Latch up immunity better than 100MeV
- SEU LET threshold better than 15 MeV*cm^2/mg
- 6 watts at 100% load
- 128Kword Data Memory
- 128Kword Program Memory

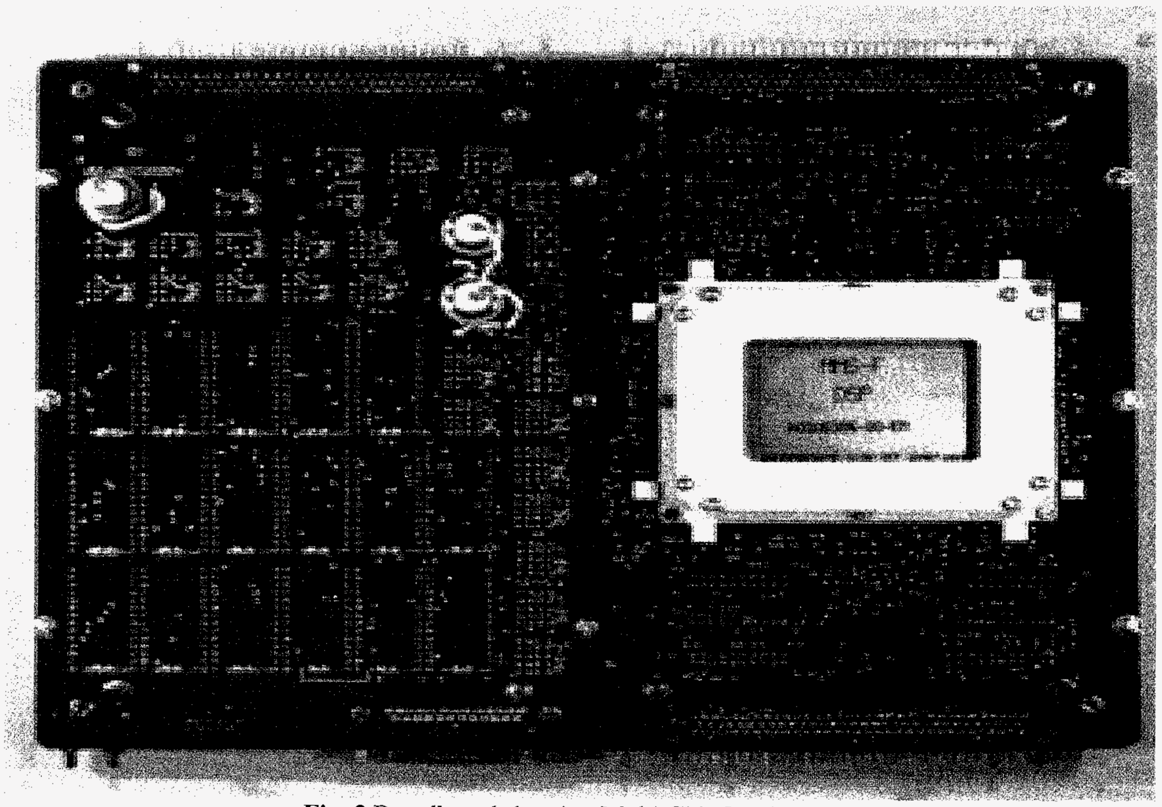


Fig. 2 Breadboard showing Multi Chip Module

4. Smcs332

The. Scalable Multi-Channel Communication Sub-System (SMCS) is a scalable and flexible high performance serial communications controller with fault tolerant features [5]. It was developed by and is heavily used by the European Aerospace community, it is incorporated into the Astrium MCMDSP hybrid and its features are listed below [6].

Main Features

- ✓ Three IEEE Std 1355-1995 links (DS-LinkTM): full duplex, up to 200 Mbit/s (in each direction) point-to-point links
- ✓ High-level packet oriented data transfer
- ✓ Autonomous command execution
- ✓ Fault tolerance features (link disconnect detection , parity check at token level, checksum generation)
- ✓ Emphasis put on high-throughput with low CPU interaction; scalable interface allows integration with any CPU type (data bus width of 8, 16 or 32 bits)
- ✓ link disconnect detection and parity check at token level
- ✓

Scalable interface to host CPU (HOCI):

- ✓ Data transfer via FIFO (internal)
- ✓ Interrupt capability for host CPU

Communication Memory Interface (COMI):

- ✓ Data transfer via DPRAM (external)
- ✓ 64 Kwords of DPRAM addressable
- ✓ Optional sign extension
- ✓ Up to two SMCS on one DPRAM (internal arbitration)

Two operation modes (per channel):

- ✓ Transparent mode:
  No interpretation of data on link, variable packet length

- ✓ SMCS protocol mode:
  Variable packet length, command execution, reset capability

Other features:

- ✓ Configuration via host CPU or over link
- ✓ Dual endian support
- ✓ Power save mode (automatic transfer rate reduction for null tokens)
- ✓ JTAG test capability (internal link loop back for test purposes)
- ✓ Checksum generation / check at packet level

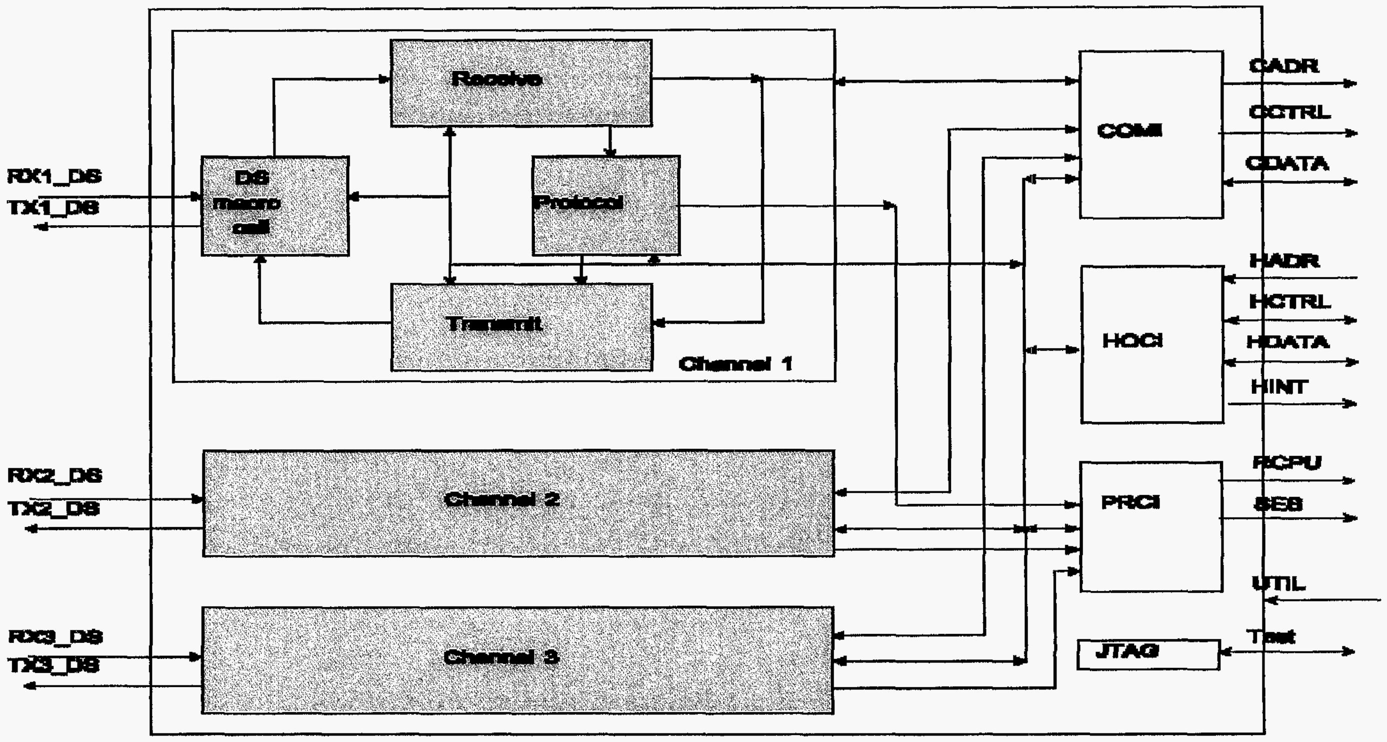


Fig 3 Scalable Multi Channel Subsystem (SMCS) 332 Block Diagram

The Astrium MCM contains the SMCS332. The SMCS contains the link layer necessary to implement the multiple links. Its associated manual described the registers used for configuring its various functions. By writing and reading to these registers via the DSP drivers could be written and called by C routines as required.

A block diagram of the SMCS is shown in Figure 3. The three channels are shown with channel one blown up to illustrate more detail.

5. SpaceWire vs Alternatives

Although selection of the MCMDSP forces us to use SpaceWire as the communication link, it turns out SpaceWire has many advantages over other communication options. Some of the previously used subsystem communication technologies implemented at the Jet Propulsion Laboratory are MIL-STD-1553, RS-422/485, Backplane Bus (VME), FireWire 1394. Each of these has its own unique advantages over the other.

The testbed requirements require multiprocessor communication among several processors sending timing, status, commands, housekeeping and science information back and forth. This must happen in real-time. Therefore the link must be fast, reliable and maintainable. The initial rate that the link must communicate at was determined to be 1.25Mb/s with the option to go faster. This made the RS-422 questionable and eliminated MIL STD-1553 technology.

IEEE 1394 FireWire

IEEE 1394 (FireWire) is a commercial technology being explored as a potential high-speed serial link for flight projects. With speeds up to 400Mb/s utilizing differential LVDS it has strong potential to be useful as a multi processor communication link. The space flight legacy on it though is nonexistent. Furthermore preliminary test have shown it to be difficult to implement due to its sophisticated exchange layer. Spacewire has a much simpler exchange layer resulting in lower gate count and higher reliability [9] not to mention a space qualified link layer already exists via the SMCS. As well as meeting the Scatterometer testbed requirements SpaceWire can be a possible candidate for competing communication technologies in future missions.

A backplane bus might be feasible for multiple processor communication. VME speeds of 64MBytes/s and. the newer VME320 backplane capable of 528Mbytes/s at 66Mhz 21 slots or 1Gbyte/s at 133 Mhz. with 10 slots [7] make VME a possible contender at a technical level. But in addition to various other limitations such as mass, power and load limitations a backplane involves greater development overhead than a comparable serial communication plan severely reducing its appeal.

The alternatives mentioned above also suffer from the commercial to flight transition problem. Often when a commercial technology is adopted to be used on flight projects it needs to be modified to overcome one shortcoming or another. By doing this adherence to the specification is broken leading to unforeseen problems, such as commercial development tools not working as intended.

A major advantage of SpaceWire is the mass and volume saving for a given bandwidth. High. mass and volume as well as power usage end up always being a concern on Spacecraft. Spacewire is far ahead over competing technologies for given performance. Implementing a fault tolerance will become easier with the simpler exchange level. Finally SpaceWire via the SMCS has a flight-qualified device not yet available from competing high-speed serial links.

6. Scatterometer And The Testbed

The SpaceWire links are part of the MCM hybrid, which is used, in the larger context of a scatterometer prototype being developed to replace the existing Seawinds project [8].

The prototype is exploring a new technology in that it will have several Digital Signal Processors configured in a multiprocessor mode used to process the return echoes from ocean waves. There will be 3 or 4 receive processors depending on the final processing load, and one transmit processor. Each receive processor will be required to establish two-way communication with the transmitter. A timing reference is transmitted to each receiver which in turn utilizes this timing data to retrieve information about the returned echo pulse. After processing the echo data the receiver sends its results back to the transmitter via SpaceWire for eventual downlink. The SpaceWire link will be used by the transmitter for sending timing data to the receiver in order to help establish pulse-to- pulse correlation, the transmitter will then receive science data back from the receivers via SpaceWire. The transmitter then forwards the data back to a Command and Data Handling System for downlink using RS 422[2].

The transmit and receive processor perform different tasks at a high level, the driver is written to support either. The higher-level software then accesses this driver. An interrupt is generated any time there is a transmission, reception or an error. The Interrupt Handler Routine then reads a register to determine if the interrupt was caused by a transmission, reception or error. Based on this information the driver software performs the tasks listed in the driver software section below.

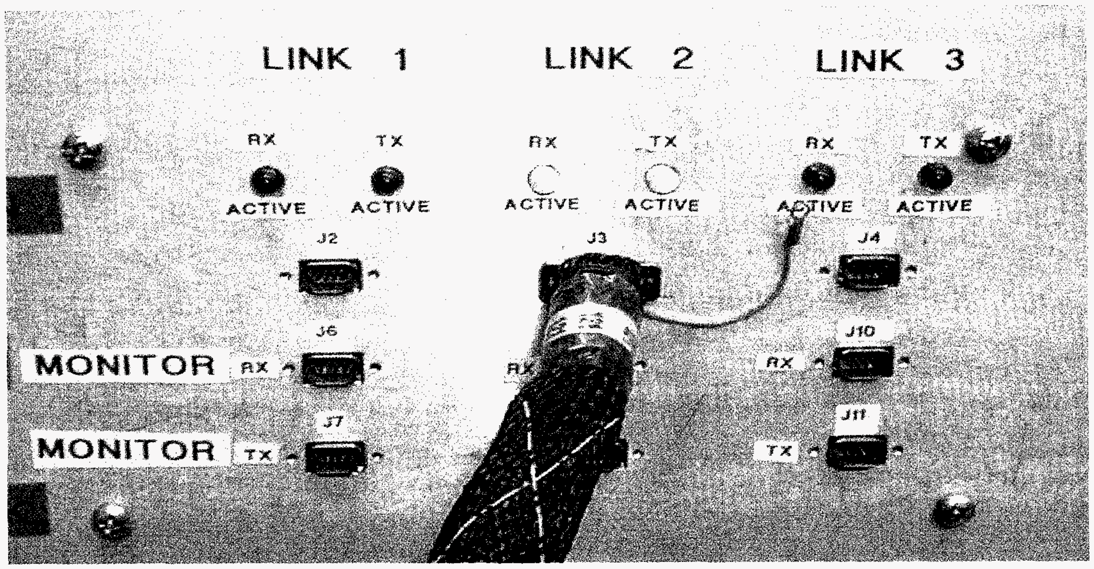


Fig.4 Front Panel with Micro D-type connectors

Fig 4 shows the Front Panel of one of the processors. Shown are the three available links, each of which contains an RX and TX for full duplex data transfer. The monitor outputs allow a test program to observe the traffic on a particular link. Connections between processors located in different chassis are made with Spacewire specified cabling. Interconnects are made with the specified Micro-D type connectors although the MCM SpaceWire port uses TTL and a standard D-Type connector.

The testbed's first attempts utilized a new Physical layer TTL to LVDS conversion board utilizing high-speed differential interface and 9 pin micro D-type connections. The initial single ended implementation led to heavy ground bounce. While a solution was attempted via implementing twisted pair cable, a temporary substitute was created. A unshielded CAT5 type cable with standard D-type connectors was used to communicate between the MCM's directly placed 5-10 feet apart using just the TTL ports. This worked fine for low speeds. On the conversion board the differential drivers and receivers used were 90C31 and 90C32. These devices have a maximum bandwidth of 155 Mbps. The interface to and from these devices to the SMCS ASIC used unbalance twisted pair with one of the signal lines in each pair connecting the signal ground on the conversion board to the signal ground on the DSP breadboard. A source termination of 200 ohms was used on the driver signals going into the DSP. Prior to doing this "ground bounce" and various fast rise time noise related problem prevented us from using the physical layer conversion board.

Standard 9 pin micro-D connectors were used to provide front panel access to the LVDS signals. Cabling required to meet the basic specification was more specialized than that used on past projects and had to be special ordered. Video/Audio[2] cable with 110 ohm characteristic impedance with both an internal shield for each twisted pair and a group shield as defined by the SpaceWire spec. was used. Figure 5 shows a completed testbed cable

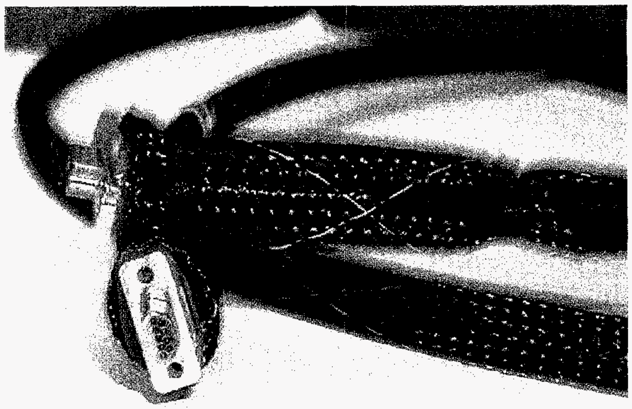


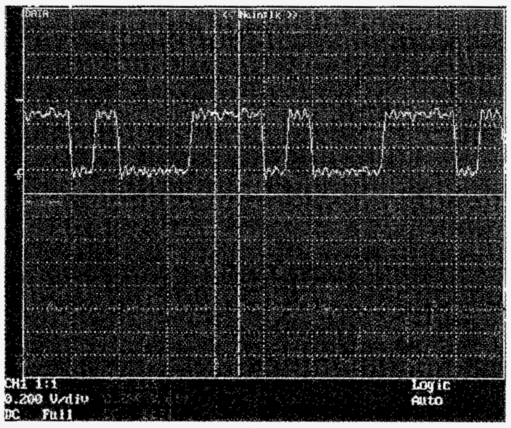
Fig. 5 Testbed SpaceWire Cable

At the same time we decided to employ LVDS as the physical layer we continued fine-tuning the driver software. This also was considered a strong contender for contributing to our problems with initial link establishment early on.

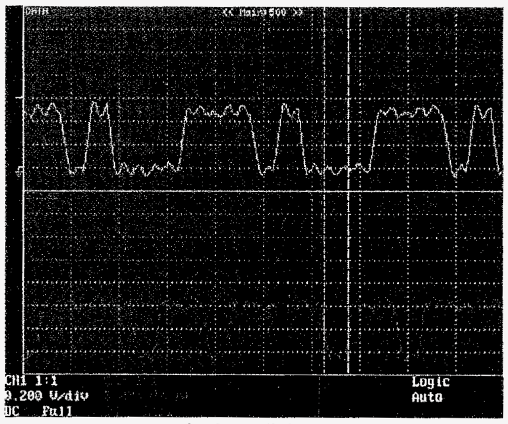
The signal characteristics at different baud rates of the new configuration are shown in Figure 6 a. thru f. These measurements were made on the MCM TTL end of the link.

[2] BELDEN Part #1803F

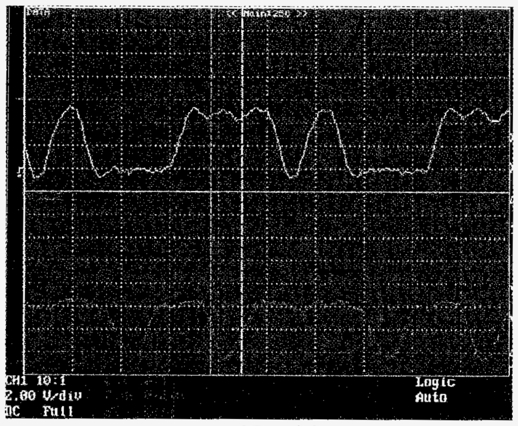
Figure 5(a.) is at 10Mb/s, (b.)20Mb/s and so in 10Mb/s with (f.) showing the 60Mb/s signal. The results indicate that further work is required on our physical layer to be able to meet the 155Mb/s speed capabilities of the SMCS, this will most likely involve high speed digital design techniques. A future plan is to repackage the DSP breadboards with the physical layer incorporated in the breadboard itself.



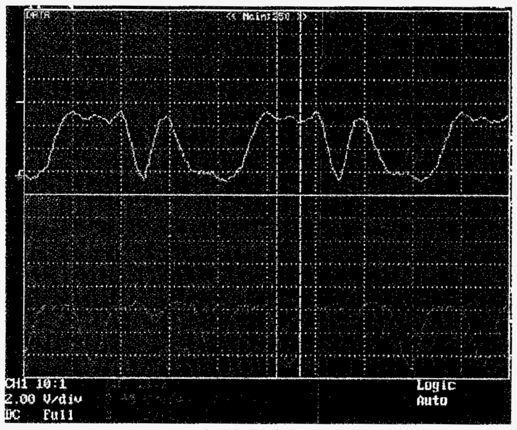
a. 10 Mb/s



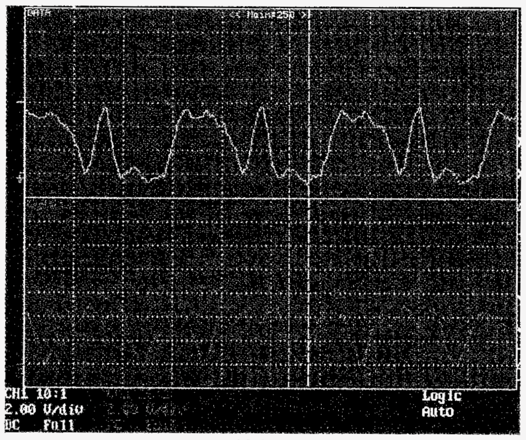
b. 20 Mb/s



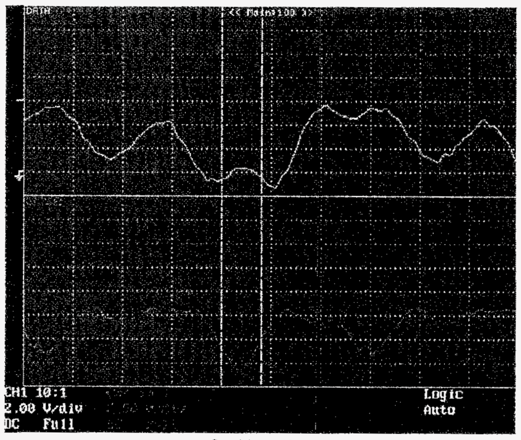
c. 30 Mb/s



d.40 Mb/s



e. 50 Mb/s



f. 60 Mb/s

FIGURE 6(a.-f.) Data and Strobe Signal shape as Baud Rate increases

SpaceWire Driver

There was a need for a driver that could be called by an application or executive program when SpaceWire activities were required. The SpaceWire driver routine on the receiver and transmitter are the same. Each is written mostly in C with time critical portions written in assembly language. It is interrupt driven for critical tasks and state driven for the rest. Runtime statistics are collected and can be enabled/disabled at compile time. It is an Application Programming Interface (API) that allows the various modes of the SMCS to be exercised. This includes

- Initializing the master
- Starting a link as master or slave
- Set baud rate
- Stopping a link

And several more functions not listed here.
The Driver API is implemented utilizing the following:

- Controlling and interfacing with the SMCS332 are through a set of 89 registers that are memory mapped to the data memory interfaced to 21020 DSP. Not all registers are used in this driver
- 32-bit Host Control Interface (HOCI), Communication Memory Interface (COMI) and transparent mode are chosen for this *implementation*
- Start up sequence are controlled by higher level application and the proper sequences as specified in IEEE-1355 are to be followed

Sending data

When a link is started and established, a flag is set to true for that link. A message is sent that the corresponding link is ready to transmit. The driver clears the flag, moves the data to dual-port ram on the SMCS and starts the transmitter, EOP is attached at the end of message. After the transmission is completed, an end-of-transmit interrupt is generated. The interrupt service routine (ISR) then resets the flag.

Receiving data

When data is received on any link, the SMCS332 puts it into dual-port ram using the preprogrammed COMI interface. Since every packet is terminated by EOP, an interrupt is generated when the EOP is received. Since the ISR is required to stay as short as possible as to not interrupt the critical timing of the echo processing, moving received data from dual-port ram to the driver's ring buffer will be done outside of the ISR. When an EOP interrupt is received, the ISR sets a flag to indicate that there is data ready to be moved, when the higher level application is done with its critical computations, it calls a function periodically and tells the driver to move data from dual-port ram to a ring buffer. The size of this ring buffer can be as large as memory allows, the default size is 8192 bytes. The ring buffer data is read first-in-first-out. When the ring buffer is full or near full and a new incoming message doesn't fit, the new message is dropped so that it won't corrupt previously received valid data in the buffer. The application can check the total number of 32-bit words currently sitting in the ring buffer by calling a function which checks to see if data is available. The function returns an integer indicating the number of unprocessed data words received. The higher level application calls another function to unload data from ring buffer to application. Another function returns the number of words successfully unloaded.

Driver Error handling

All links were programmed to auto-restart when initialized. A parity/disconnect error generates an interrupt. A disconnect/parity can happen during the transmission or receipt of a message/packet. By default, the transmit section is reset by the SMCS after a disconnect error has occurred, the receive section is not. The ISR therefore rearms/restarts the receiver and new transmit/receive transfers can be started.

Transfer results

Figure 7 is a graph showing the end results of our software and hardware, it shows the measured disconnect errors reported by the SMCS as the baud rate is increased. The greatest contribution to these errors is most likely a result of our physical layer conversion board. As can be seen from the signal characteristics in Figure5 a.-f. there is significant degradation of signal quality with baud rate increase. The link could not be reliably be maintained above 60Mb/s.

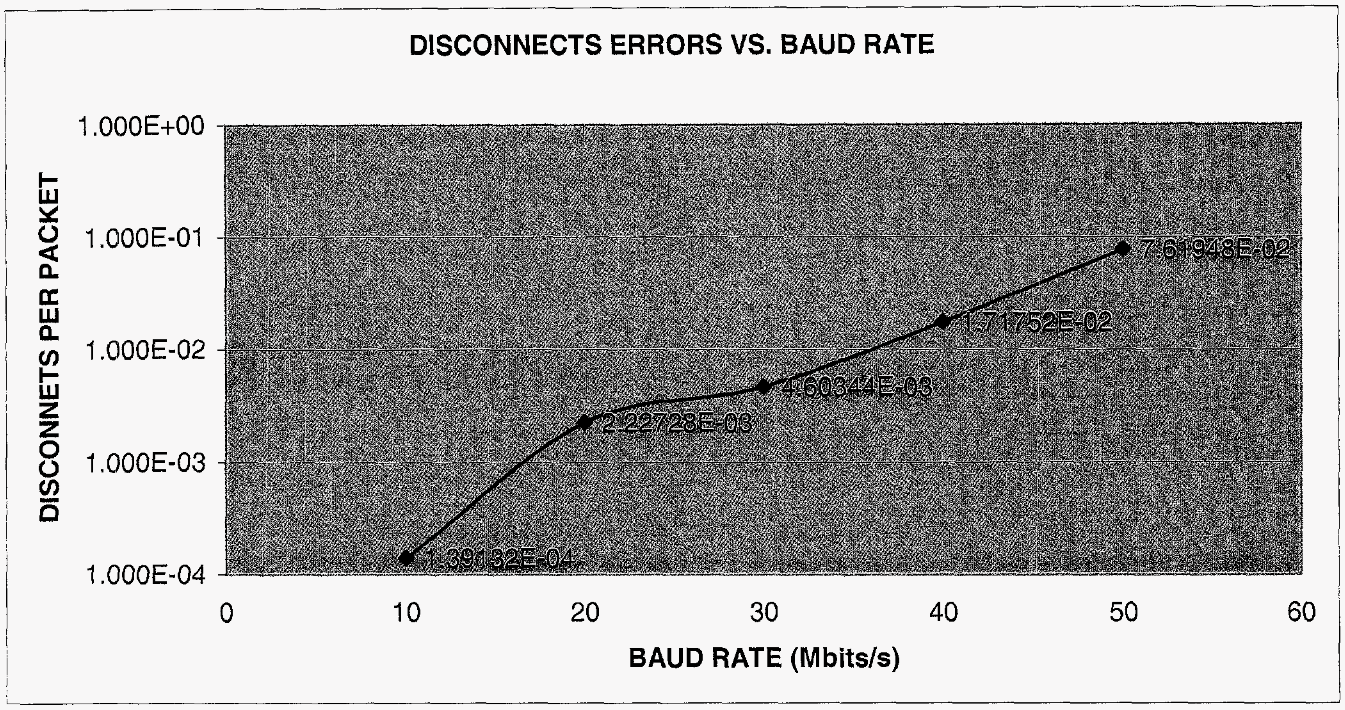


Fig. 7 Disconnect errors vs. baud rate for testbed MCM implementation

7. Testing SpaceWire

Although the initial main test of our SpaceWire link was simply, to determine if it works or not, we have shown that we can keep errors down while transmitting our data. A more thorough test will have to be developed to evaluate its performance under margin, stress, failure, and anomalies. Especially when this evolves to a flight project.

A LABVIEW application has been written that utilizes a commercial PCI SpaceWire card from 4-Links and included associated driver software to support interfacing to the SMCS. By using local loop back tests using two ports on the card initial task were to establish communication between the ports. This verified our cabling construction as well exercising our knowledge of the signaling protocol. This setup was then used to interface to the SMCS and confirm two-way communication between different systems. A driver software configuration anomaly was found this way. Future use will include using the LABVIEW program as a monitor. Part of the SpaceWire driver software function will record values such as

- Number of Packets received
- Number of Packets sent.
- Number of Parity errors
- Number of dropped links
- Total words transmitted

This data will be used to characterize the quality of the SpaceWire link as well as help troubleshoot science data inconsistencies.

8. Conclusion

By adopting an MCMDSP with SpaceWire technology we are introduced into a new method of transferring serial data from one subsystem to another. SpaceWire has been proven to work reliably, is easy to implement, has a flight legacy and proven to be a more cost-effective high performance solution than competitive technology such as FireWire 1394, RS-422/485 Mil 1553 or a bus. The existence of rad-tolerant hardware enormously increases its appeal. There is strong potential for SpaceWire on future flight missions.

Acknowledgements

The work performed in this paper was carried out at the Jet Propulsion Laboratory, California Institute of Technology, under a contract with the National Aeronautics and Space Administration.

We would also like to thank Astrium, specifically Pierre-Eric Berthet for the extended loan of the MCMDSP hybrids on the evaluation boards.

References

[1] M.SPENCER, et al," Improved Resolution Backscatter Measurements with the Seawinds Pencil-Beam Scatterometer" *IEEE Transactions on GeoScience and Remote Sensing* Vol. 35, No.1, January 200

[2] D.CLARK, J.LUX, M.SHIRBACHEH, "Testbed for Development of a DSP-Based Signal Processing Subsystem for an Earth-Orbiting Radar Scatterometer" *submitted for presentation at IEEE Aerospace Conference*, Big Sky Montana, March 2002

[3] http://www.1355-association.org/

[4] ECSS-E-50-12 DRAFT 1, "SpaceWire-Links, Nodes, Routers, and Networks", *European Cooperation for Space Engineering*, March.2001

[5] "SMCS332 USERS MANUAL ISSUE 2", *DSS*, DIPSAPII, OMI April 04 1999

[6] http://www.estec.esa.nl/tech/spacewire/

[7] M HECKMAN, D BERDING, "Using VME320 back plane technology to speed up your VMEbus system", *VMEbus Systems,* 26-29, August 2001

[8] A. BACHMANN, et al," Multiprocessor Digital Signal Processing on Earth Orbiting Scatterometers", *IEEE Aerospace Conference*, Big Sky Montana, March 2001

[9] S.CHAU "A Design-Diversity Based Fault-Tolerant COTS Avionics Bus Network" *submitted for presentation at the Pacific Rim Dependable Computing Conference, South Korea Dec.17-19, 2002*

Alex Bachmann has been involved in Spacecraft and Ground System Test and Integration for 15 years. His interests at JPL include developing test methodologies for command and data handling systems as well as advancing DSP and JTAG technology on space projects.

Minh Lang *is a member of the technical staff in JPL's Flight Systems Section. He earned a Bachelor's degree in Computer Science from California State University, Northridge. He has been involved in design and development of embedded software in JPL since 1988.*

James Lux *is in the Spacecraft Telecommunications Section at JPL. Scatterometers design is only one of Jim's diverse interests, which range from man-made tornadoes to solar eclipses to high performance digital radio links.*

Richard Steffke *graduated California State Polytechnic University Pomona BA Aerospace Engineering. He was lead test engineer on NSCAT DSS, which was the first scatterometer to fly a DSP.*